מבני נתונים, סמסטר א' תש"פ
המכללה האקדמית של תל-אביב-יפו

תרגיל תכנות 1
נושא התרגיל: יציאה ממבוך ויצירה של מבוך

תאריך הגשה: יום חמישי ה- 19.12.19 בחצות.

הנחיות כלליות

א. התרגיל הינו תרגיל חובה.
ב. התרגיל ניתן להגשה בבודדים או בזוגות, אך לא בקבוצות גדולות יותר.
ג. גם תלמידים החוזרים על הקורס מחויבים בהגשת התרגיל, ולא משנות הנסיבות.
ד. ציון "נכשל" בתרגיל, או הגשה מאוחרת שלו ללא אישור מתאים שניתן על ידי המרצה, יגררו ציון "נכשל" סופי בקורס.
ה. איחור בהגשה יאושר רק במקרה של מילואים, מחלה או לידה, וגם זאת רק בתנאי שהפנייה למרצה בנושא נעשתה לפני מועד ההגשה המקורי של התרגיל.

מטרת התרגיל

עליכם לכתוב תוכנית בשפת C++ (ובה בלבד. בחירת השפה אינה נתונה לשיקול דעת הסטודנט), המבצעת שתי מטרות מרכזיות:

1. יציאה ממבוך נתון.
2. יצירה של מבוך מקרי.

השלבים לביצוע התרגיל

א. קראו תחילה היטב את ההנחיות של התרגיל והבינו את האלגוריתמים המתוארים בו. מומלץ להריץ את האלגוריתמים "הרצה יבשה" על דוגמאות.
ב. שימו לב, פתרונות אשר אינם עושים שימוש במחסנית ובתור כפי שמתוארים בתרגיל או מציעים אלגוריתמים חלופיים לאלה שמתוארים כאן ייפסלו על הסף!
ג. תכננו את ה-design של התוכנית שלכם: בחרו אילו מחלקות תממשו, החליטו על data members מתאימים ועל methods רלוונטיים לכל מחלקה.
ד. כתבו מימוש מלא לכל המחלקות שייעשה בהן שימוש במסגרת התוכנית.
ה. אסור השימוש במחלקות stack, list, queue ו- dequeue של STL אך מותר להשתמש ב- vector.

תיאור האלגוריתמים ומטרת התרגיל:

תתבקשו לכתוב פונקציה שיוצאת ממבוך תוך שימוש בתור ופונקציה שיוצרת מבוך תוך שימוש במחסנית.

מבוך הוא פשוט מערך דו-מימדי של קירות ומקומות פנויים, כאשר בין כל מקום פנוי נתון יש או אין קיר.

נייצג מקום פנוי על ידי רווח וקיר על ידי *, כאשר המבוך כולו מוקף על ידי קירות, פרט לפינה של הכניסה והפינה של היציאה.

ההתחלה של המבוך תהיה בפינה השמאלית העליונה והיציאה של המבוך תהיה בפינה הימנית התחתונה.

למבוך יש כמובן גובה ורוחב (h,w), כאשר המשבצת העליונה השמאלית נמצאת במקום (0,0) והמשבצת התחתונה הימנית נמצאת במקום (h-1, w-1).

במבוך תקין השורה התחתונה, השורה העליונה, העמודה השמאלית והעמודה הימנית יהיו תמיד קירות שמיוצגים על ידי כוכביות, למעט הכניסה והיציאה מהמבוך.

נאמר ששתי משבצות במבוך **"נגישות"** אם הם שכנות במבוך ונמצאות באותה שורה או עמודה, ואין ביניהן קיר. שימו לב שאסור ללכת באלכסון, אלא רק לארבעה כיוונים (שמאלה, ימינה, למעלה, למטה), כמובן רק אם אין שם קיר.

כך למשל יכול מבוך להיראות בצורה הזאת:

```
* * * * * * * * * * *
*       *       *   *   כניסה למבוך
*   * * * *   *   *   *
*               *   *
*   * * * *   * * *   *
*   *       *       *   *
*   *   *   * * *   *   *
*       *   *       *   *
*   *   * * * *   * * *
*                       יציאה ממבוך
* * * * * * * * * * *
```

פתרון מבוך על ידי תור:

פתרון של מבוך פירושו לחפש את נקודת היציאה כשאנחנו מתחילים בנקודת הכניסה. כדי להבטיח שנמצא את היציאה (אם יש מסלול אליה), עלינו לוודא שאנחנו עוברים על כל המשבצות האפשריות במבוך. התור יאפשר לנו לזכור היכן כבר ביקרנו ולהחליט מה המשבצת הבאה שבא נבקר. האלגוריתם הבסיסי שבו נשתמש הוא:

1. נאתחל את התור במשבצת (1,0).
2. כל עוד התור לא ריק:

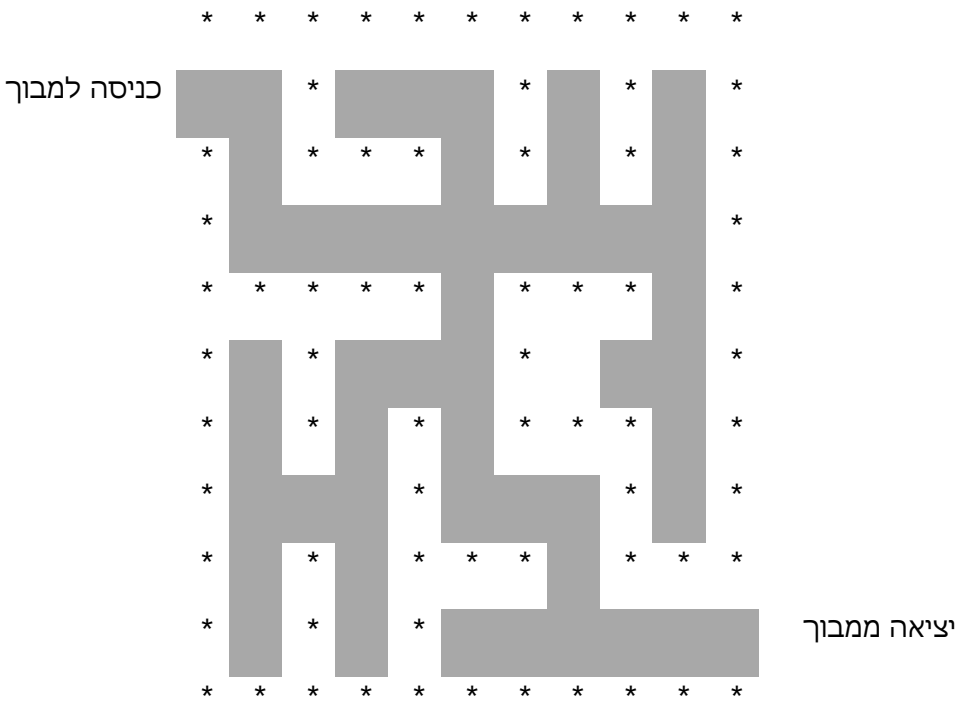
- נוריד מראש התור מיקום של משבצת ונסמן שביקרנו בה.
- אם המשבצת הנוכחית שהורדנו היא נקודת הסיום אז נצא מהלולאה כי יצאנו מהמבוך.
- אחרת: נוסיף לתור את כל המשבצות שאליהן אפשר להגיע מהמשבצת הנוכחית ושעדיין לא ביקרנו בהן ונמשיך בלולאה הראשית.

שימו לב שהתור מכיל מיקומים של משבצות ריקות שמכילות רווחים, ובכל פעם שמורידים משבצת מהתור אנחנו בודקים את השכנים שמסביבה שאפשר להגיע אליהם ואז מוסיפים אותם לתור (אם עדיין לא היינו בהם) כדי לבדוק אותם בשלב מאוחר יותר.

בגלל שהתור פועל בצורה של FIFO אז צורת החיפוש תיראה כמו מים שנשפכים מנקודת ההתחלה (ייתכן שדרך כל המבוך), עד שמגיעים לנקודת הסיום.

<u>כדי לאפשר בדיקה אחידה של התרגילים שלכם הקפידו לבקר בשכנים בסדר הבא: ימינה, למטה, שמאלה, למעלה.</u>

הפלט שלכם במקרה של פתרון מבוך לא יהיה רק המסלול שמוביל ליציאה, אלא כל המבוך שבו מסומנות **כל** המשבצות שביקרתן בהן עד שמצאתם את היציאה **(את המשבצות שבהן ביקרתם סמנו בסימן $).** למשל במבוך שבדוגמה האלגוריתם יבקר בכל המשבצות לפני שיגיע ליציאה, אבל זה לא חייב להיות כך ובהחלט ייתכן שהאלגוריתם יעצור לפני שביקר בכל המשבצות (ברגע שהגיע ליציאה). המשבצות שבהן האלגוריתם ביקר בדוגמה סומנו בצבע אפור ולא ב- $, אולם כאמור, אתם תסמנו בפתרון שלכם את המשבצות שבהן ביקרתם ב- $.



<u>יצירת מבוך:</u>

יצירת מבוך דומה מאוד לפתרון מבוך, אבל הפעם המטרה היא לבקר בכל המקומות הריקים ולהסיר קירות במהלך הביקור, כך שייווצר מסלול בין כל שתי משבצות ריקות על ידי סדרה של משבצות נגישות. זה כמובן יבטיח גם שיהיה מסלול מהכניסה ליציאה של המבוך. שימו לב, אסור שתישאר בסוף סדרה של משבצות ריקות שאי אפשר להגיע עליהן בדרך כלשהי (כלומר סדרה של משבצות ריקות מוקפות לגמרי על ידי קירות). הנה האלגוריתם ליצירת מבוך:

1. נתחיל ממבוך שבו כל הקירות קיימים, בין כל שתי משבצות סמוכות, וכמובן הקירות מסביב למבוך למעט הכניסה (1,0) והיציאה (h-2,w-1).
2. נאתחל את המחסנית עם המשבצת (1,1).
3. כל עוד המחסנית לא ריקה:

- נוריד מראש המחסנית מיקום של משבצת ונסמן שביקרנו בה.
- אם למשבצת הנוכחית שהורדנו יש שכנים שעדיין לא ביקרנו בהם אז:
  א. נבחר שכן **<u>מקרי</u>** של המשבצת שעדיין לא ביקרנו בו.
  ב. נוריד את הקיר בין השכן שנבחר והמשבצת הנוכחית.
  ג. נחזיר את המשבצת הנוכחית למחסנית.
  ד. נשים גם את השכן המקרי שנבחר במחסנית.
- נמשיך בלולאה הראשית.

למעשה, האלגוריתם הזה יבקר במבוך כמו נחש שזוחל עד שהוא מגיע למקום ללא מוצא (שזה משבצת בלי שכנים שעדיין לא ביקרנו בהם), ואז הנחש חוזר אחורה עד שהוא מוצא משבצת שיש לה עדיין שכנים שבהן היא לא ביקרה וממשיך שוב קדימה.

נסו להריץ את האלגוריתם על המבוך ההתחלתי הבא שמכיל את כל הקירות ההתחלתיים כדי להבין איך האלגוריתם עובד.

```
*  *  *  *  *  *  *  *  *  *  *
      *     *     *     *     *
*  *  *  *  *  *  *  *  *  *  *
*     *     *     *     *     *
*  *  *  *  *  *  *  *  *  *  *
*     *     *     *     *     *
*  *  *  *  *  *  *  *  *  *  *
*     *     *     *     *     *
*  *  *  *  *  *  *  *  *  *  *
*     *     *     *     *
*  *  *  *  *  *  *  *  *  *  *
```

הערות חשובות:

1. בחירת השכן המקרי תעשה על ידי שימוש בפונקציה rand (קראו איך להשתמש בה).
2. <u>שימו לב גם כאן לבקר בשכנים של משבצת לפי הסדר הבא: ימינה, למטה, שמאלה, למעלה.</u>

<u>גודל המבוך המקסימלי יהיה 25 שורות ו-80 עמודות (כולל קירות), וזה כדי שאפשר יהיה להדפיס את המבוך על המסך.</u>

<u>מבני הנתונים</u>

לצורך ביצוע התרגיל, הנכם <u>נדרשים</u> לממש את המחלקות הבאות ללא שימוש ב- STL .

<u>תור Queue</u> שימומש על ידי מערך מעגלי כפי שלמדנו בכיתה, כאשר יעילות כל פעולה היא O(1). מכיוון שהתור ממומש במערך (מעגלי) אז כמות הזיכרון בו מוגבלת כמובן. לכן חשבו מראש מה הגודל המקסימלי שעליכם להקצות לתור. גודל התור ייקבע רק לאחר קריאת הקלט.

<u>מחסנית Stack</u> שתמומש על ידי רשימה מקושרת חד-כיוונית כאשר יעילות כל פעולה היא O(1). המחסנית אינה מוגבלת במקום.

ודאו שכל אחת מהמחלקות האלה מכילה את כל הפעולות הבסיסיות שהוגדרו בכיתה על מחסנית/תור.

<u>כמו כן עליכם לממש כמובן כל מחלקה אחרת שלה תזדקקו במהלך התכנית.</u>

<u>התכנית הראשית :</u>

ה- main של התכנית יבצע כמה דברים:

1. ישאל את המשתמש האם הוא רוצה להכניס מבוך משל עצמו לפתרון או שלייצר לו מבוך מקרי בגודל שהמשתמש יבחר.
2. אם המשתמש בחר בהכנסת מבוך אז התכנית תקרא את הקלט באופן הבא:
הקלט יתחיל בקליטת שני מספרים, מספר שורות ומספר עמודות (בסדר הזה),
ואז ייקרא כמחרוזות. כל מחרוזת תייצג שורה ולכן כל המחרוזות תהיינה באורך זהה שלא יעלה על 80 תווים. מחרוזת תורכב מכוכביות ורווחים בלבד. זכרו לבדוק אחרי קריאת הקלט שקיבלתם מבוך תקין (למשל, כניסה ויציאה במקום הנדרש, קירות סביב המבוך).
3. אם המשתמש בחר בייצור מבוך מקרי אז התכנית תבקש מהמשתמש את מידות המבוך (מספר שורות ומספר עמודות) שבהן הוא מעונין ותייצר מבוך מקרי בגודל המתאים.
4. התכנית תפתור את המבוך שיש לה (זה שהוזן על ידי המשתמש או המבוך המקרי שהתכנית עצמה ייצרה), ותדפיס כפלט את המבוך הפתור עם **כל המשבצות** שבהן ביקר האלגוריתם (וביניהן כמובן המשבצות של המסלול מהכניסה ליציאה, אם יש כזה מסלול).

הנחיות הגשה

יש להגיש במערכת mama במקום המיועד להגשה את הקבצים הבאים:

1. קובץ readme שיכיל את כל פרטי ההגשה הבאים:

כותרת – תרגיל תכנות מס' 1 במבני נתונים תש"ף סמסטר א'.

שורה מתחת - שמות המגישים, מספרי ת.ז. שלהם ומספר הקבוצה של כל אחד מהם (מותר להגיש עם בן זוג מקבוצה אחרת).

שימו לב: קובץ טקסט פשוט – לא word.

2. כל קבצי הקוד בסיומות cpp. ו- h.

שימו לב! הגשה שאינה בפורמט הנדרש תידחה אוטומטית.

בדיקת שגיאות:

הקפידו לבדוק שגיאות אפשריות בקלט. כך למשל שגיאות יכולות להכיל שורות לא באורך זהה בקלט, מבוך גדול מדי, אין מסגרת סביב המבוך, תווים שאינם רווח או כוכבית ועוד. במקרה של שגיאה כתבו הודעת שגיאה למסך invalid input וצאו מהתכנית באמצעות הפונקציה exit(1)

(יש לבצע #include <stdlib.h> על מנת להשתמש בה).

הערות הכרחיות נוספות

- הקפידו על תיעוד, שמות בעלי משמעות למשתנים, מודולאריות וכל מה שנדרש מתכנות נכון. בתיעוד בראש התוכנית כתבו גם הוראות הפעלה מדויקות וברורות.
- הקפידו על חלוקה נכונה לקבצים (קבצי CPP וקבצי H לכל מחלקה).
- תכנתו Object Oriented והימנעו מאלמנטים מיותרים של קוד פרוצדוראלי.
- הקפידו לשחרר את כל הזיכרון אשר הקציתם דינמית לאחר שהשתמשתם בו.
- במקרה של תקלה בריצת התוכנית (מסיבה כלשהי), עליה לדווח זאת למשתמש טרם סיימה לרוץ.
- בדקו את תוכניותיכם על קלטים רבים ככל האפשר - כולל קלטים חוקיים ולא חוקיים, וזאת בנוסף לטסטים שהוכנו עבורכם במאמא.
- תנאי הכרחי (אך כמובן לא מספיק) לקבלת ציון עובר על התרגיל, הוא שהקוד יעבור קומפילציה. ציונו של תרגיל אשר אינו עובר קומפילציה יהיה 0.

המשך סמסטר מהנה,
צוות מבני נתונים, סמסטר א' תש"ף